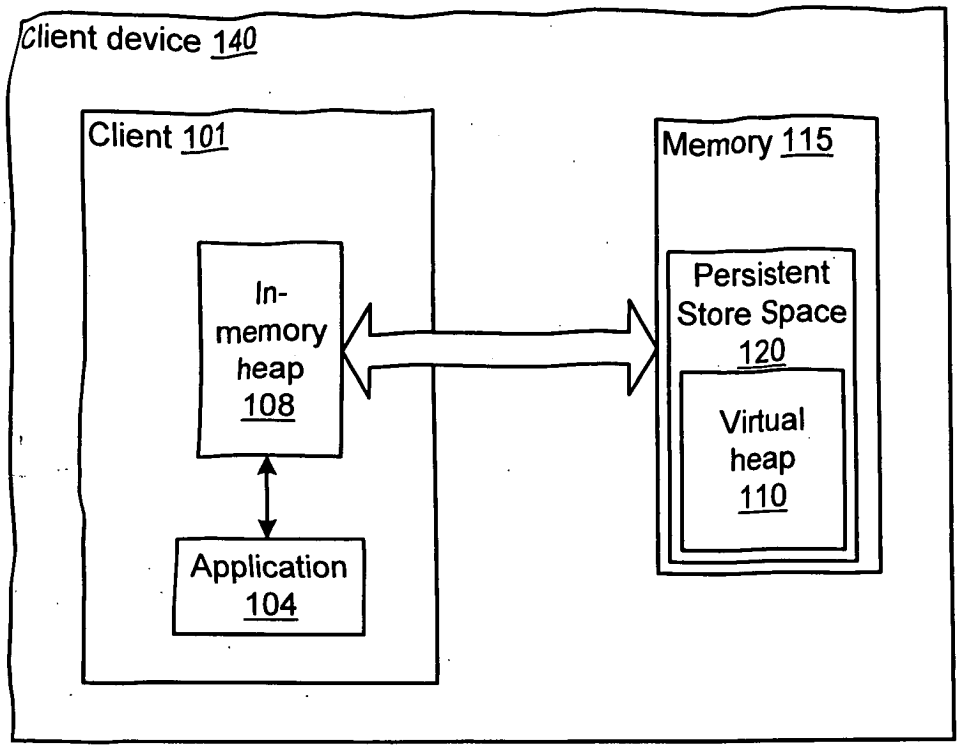

Figure 1a

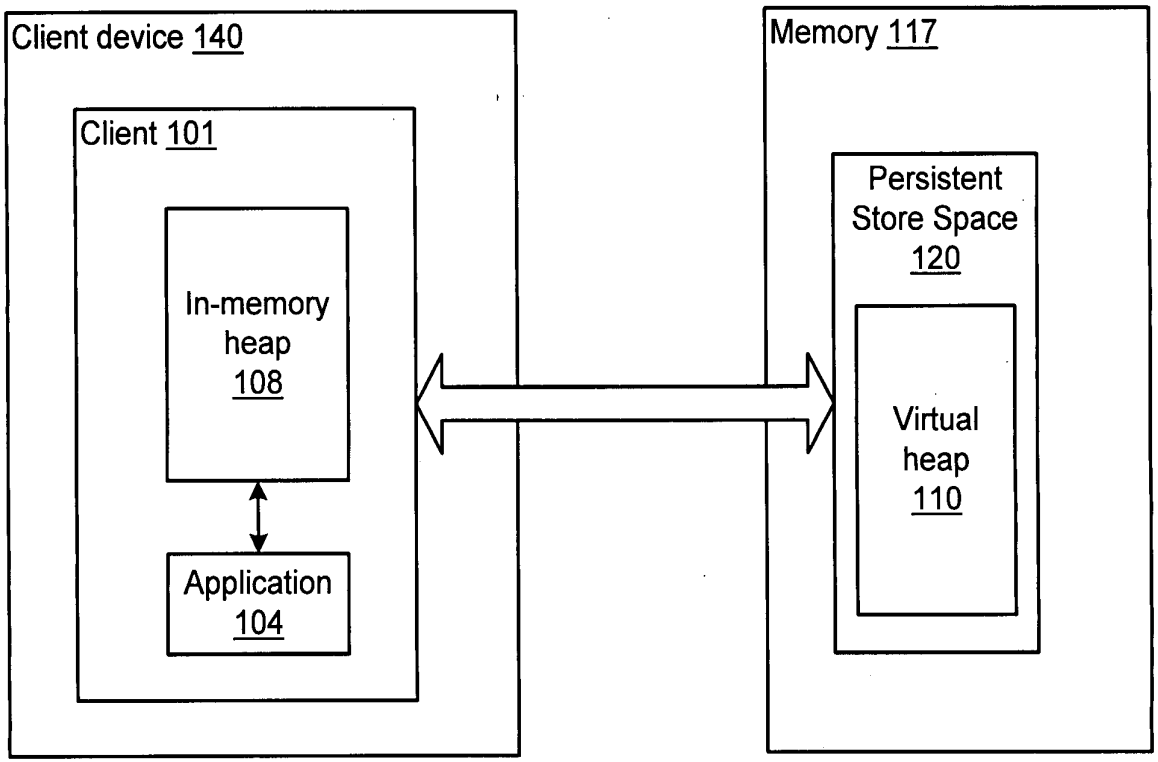
Client device 140
Client 101
In-memory heap 108
Application 104
Memory 117
Persistent Store Space 120
Virtual heap 110

Figure 1b

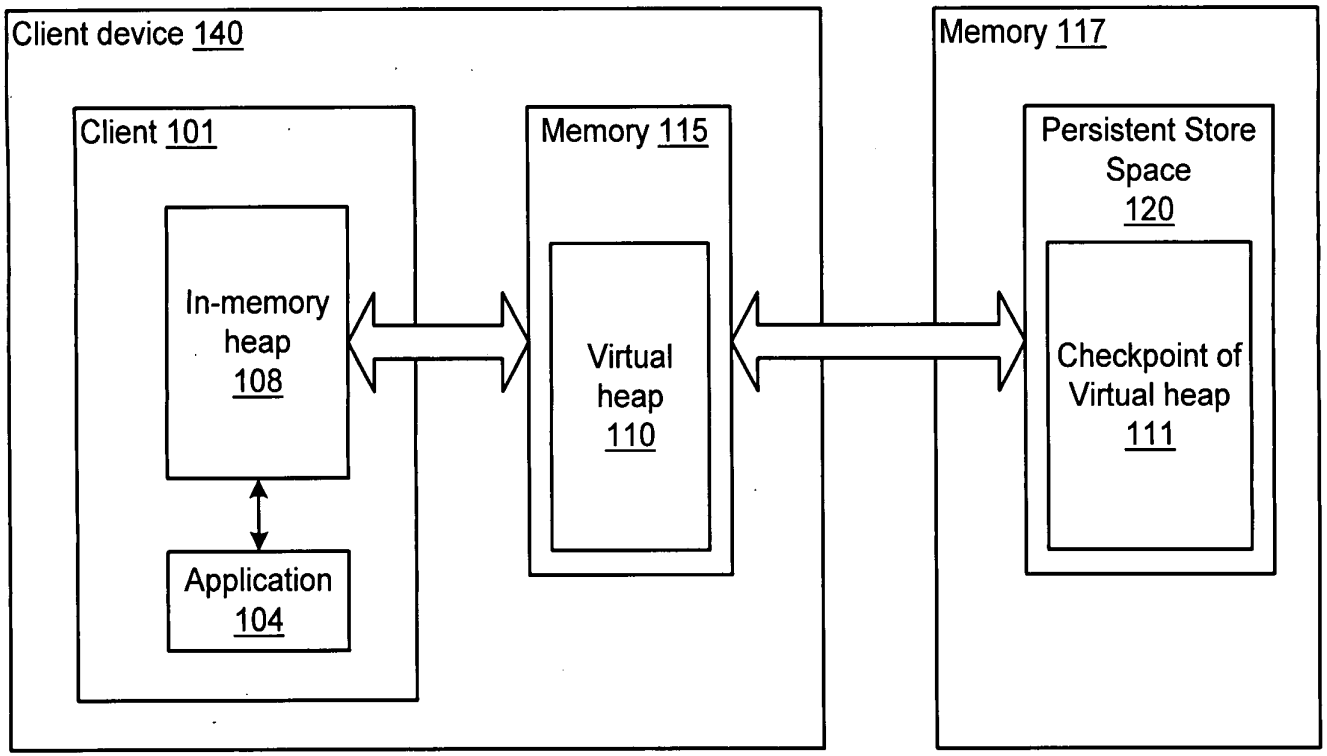
Client device 140
Client 101
In-memory heap 108
Application 104
Memory 115
Virtual heap 110
Memory 117
Persistent Store Space 120
Checkpoint of Virtual heap 111

Figure 1c

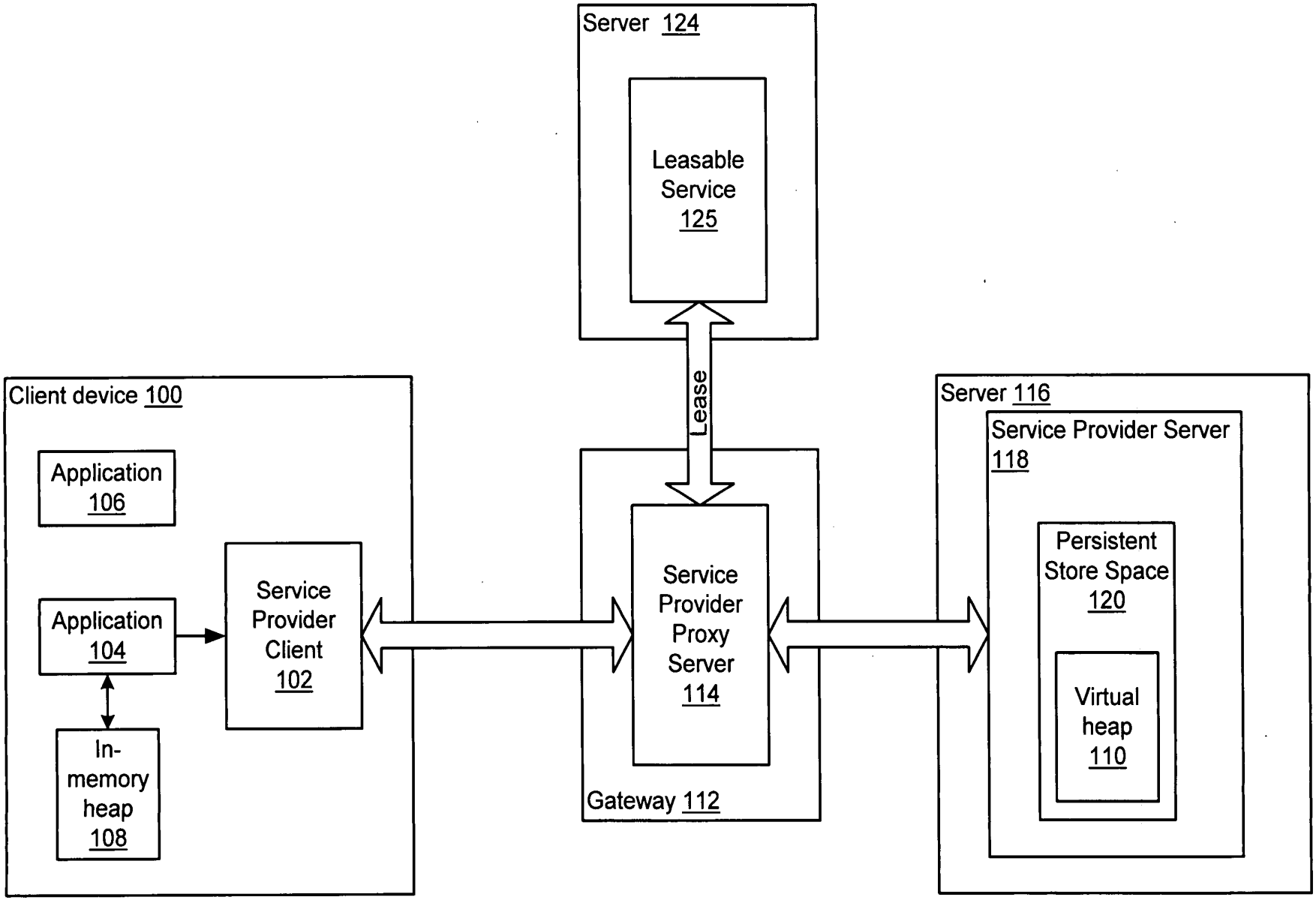
Server 124
Leasable Service 125
Lease
Client device 100
Application 106
Application 104
Service Provider Client 102
In-memory heap 108
Service Provider Proxy Server 114
Gateway 112
Server 116
Service Provider Server 118
Persistent Store Space 120
Virtual heap 110

Figure 1d

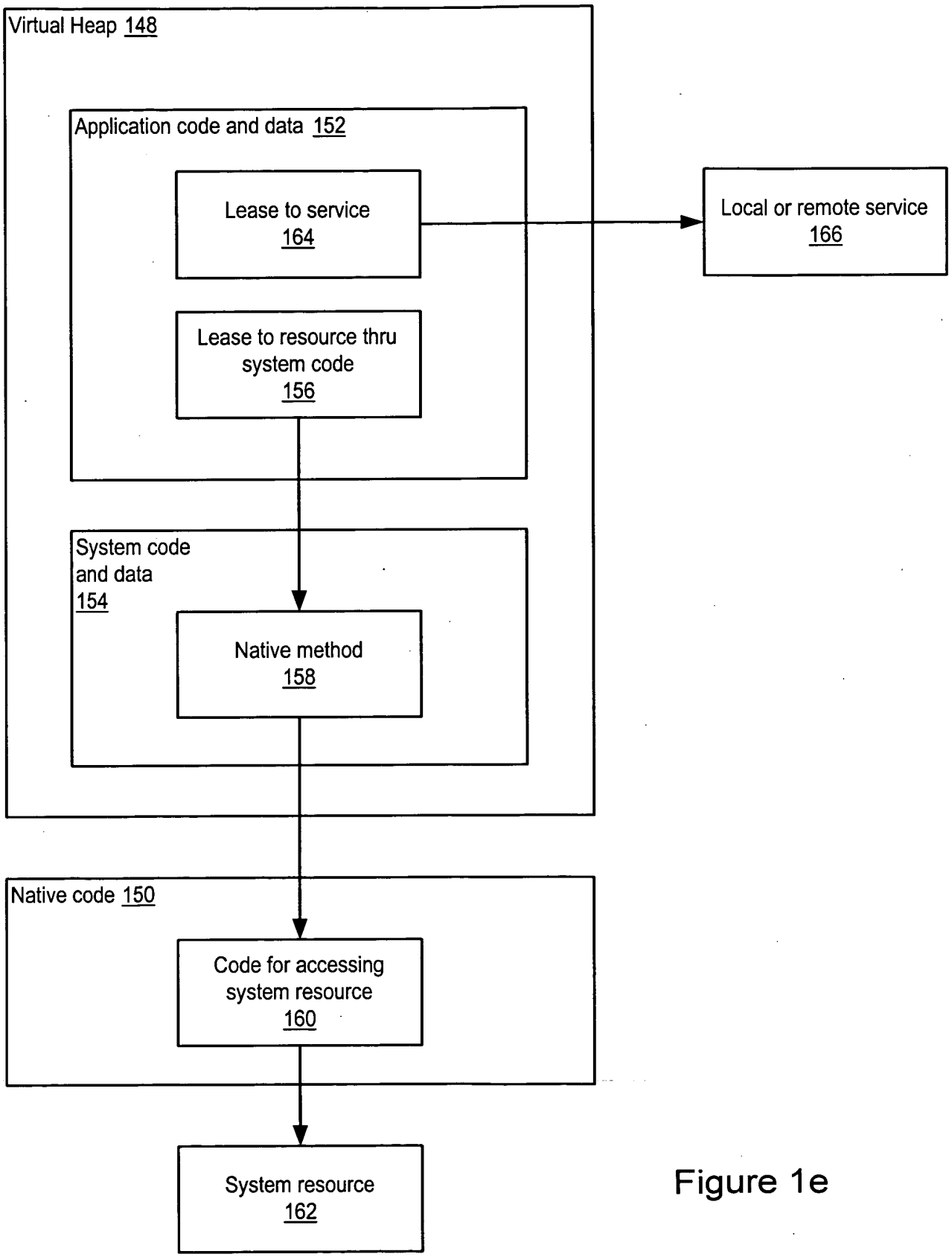
Virtual Heap 148
Application code and data 152
Lease to service 164
Local or remote service 166
Lease to resource thru system code 156
System code and data 154
Native method 158
Native code 150
Code for accessing system resource 160
System resource 162

Figure 1e

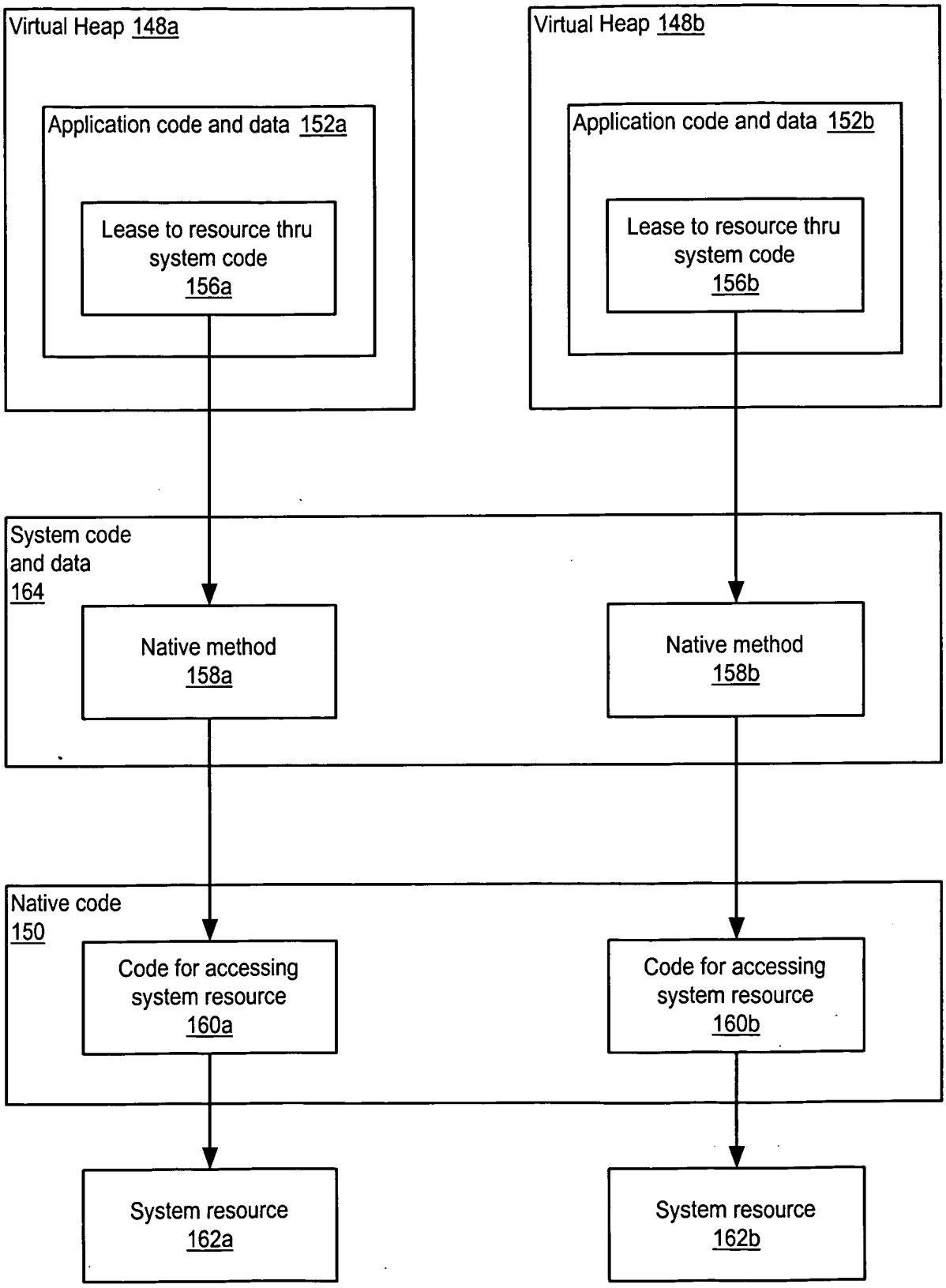
Virtual Heap 148a
Application code and data 152a
Lease to resource thru system code 156a
Virtual Heap 148b
Application code and data 152b
Lease to resource thru system code 156b
System code and data 164
Native method 158a
Native method 158b
Native code 150
Code for accessing system resource 160a
Code for accessing system resource 160b
System resource 162a
System resource 162b

Figure 1f

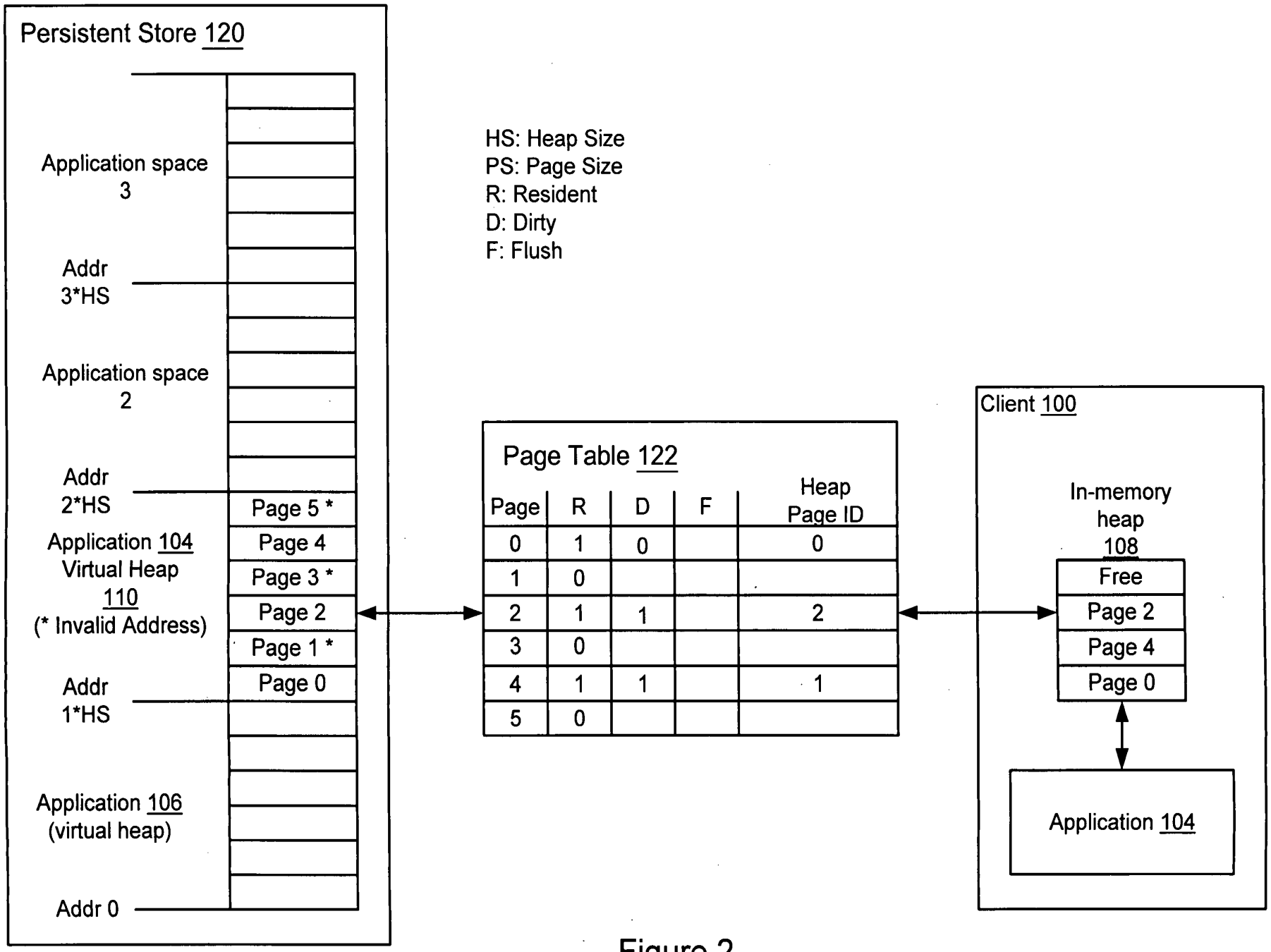
Persistent Store 120
Application space 3
Addr 3*HS
Application space 2
Addr 2*HS
Application 104 Virtual Heap 110 (* Invalid Address)
Addr 1*HS
Application 106 (virtual heap)
Addr 0
Page 5 *
Page 4
Page 3 *
Page 2
Page 1 *
Page 0
HS: Heap Size
PS: Page Size
R: Resident
D: Dirty
F: Flush
Page Table 122
Page | R | D | F | Heap Page ID
0 | 1 | 0 | | 0
1 | 0 | | |
2 | 1 | 1 | | 2
3 | 0 | | |
4 | 1 | 1 | | 1
5 | 0 | | |
Client 100
In-memory heap 108
Free
Page 2
Page 4
Page 0
Application 104

Figure 2

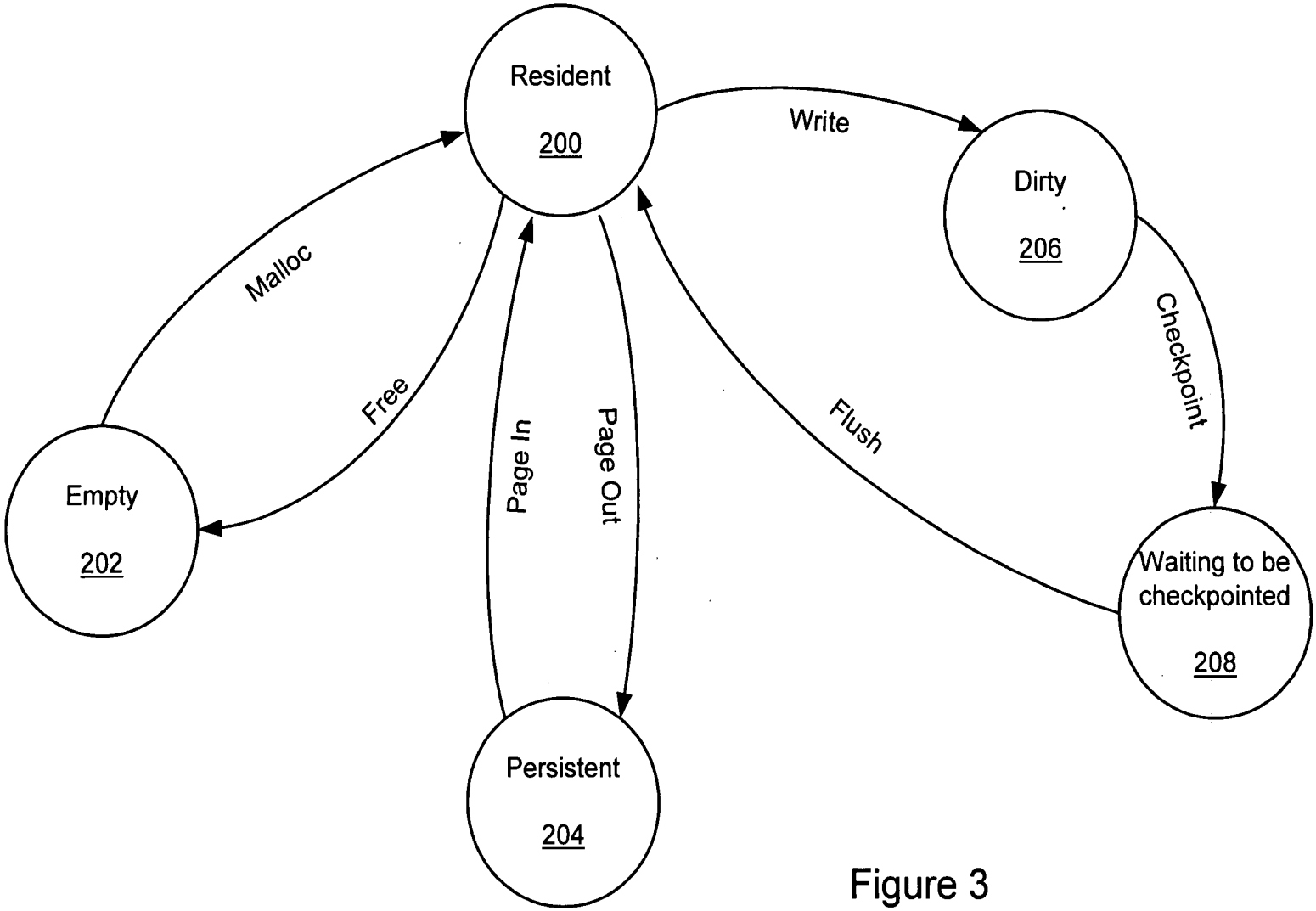
Resident
200
Write
Dirty
206
Malloc
Free
Checkpoint
Flush
Page In
Page Out
Empty
202
Waiting to be checkpointed
208
Persistent
204

Figure 3

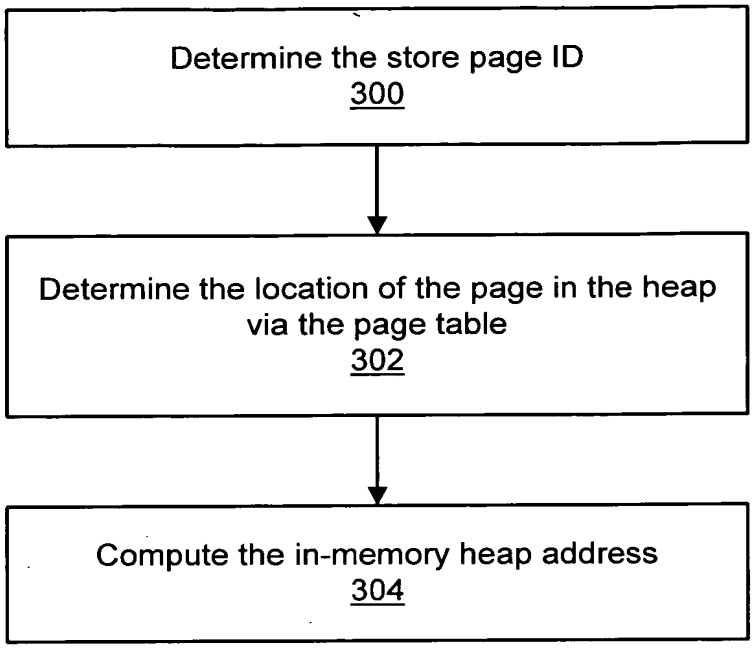
Determine the store page ID
300
Determine the location of the page in the heap via the page table
302
Compute the in-memory heap address
304

Figure 4

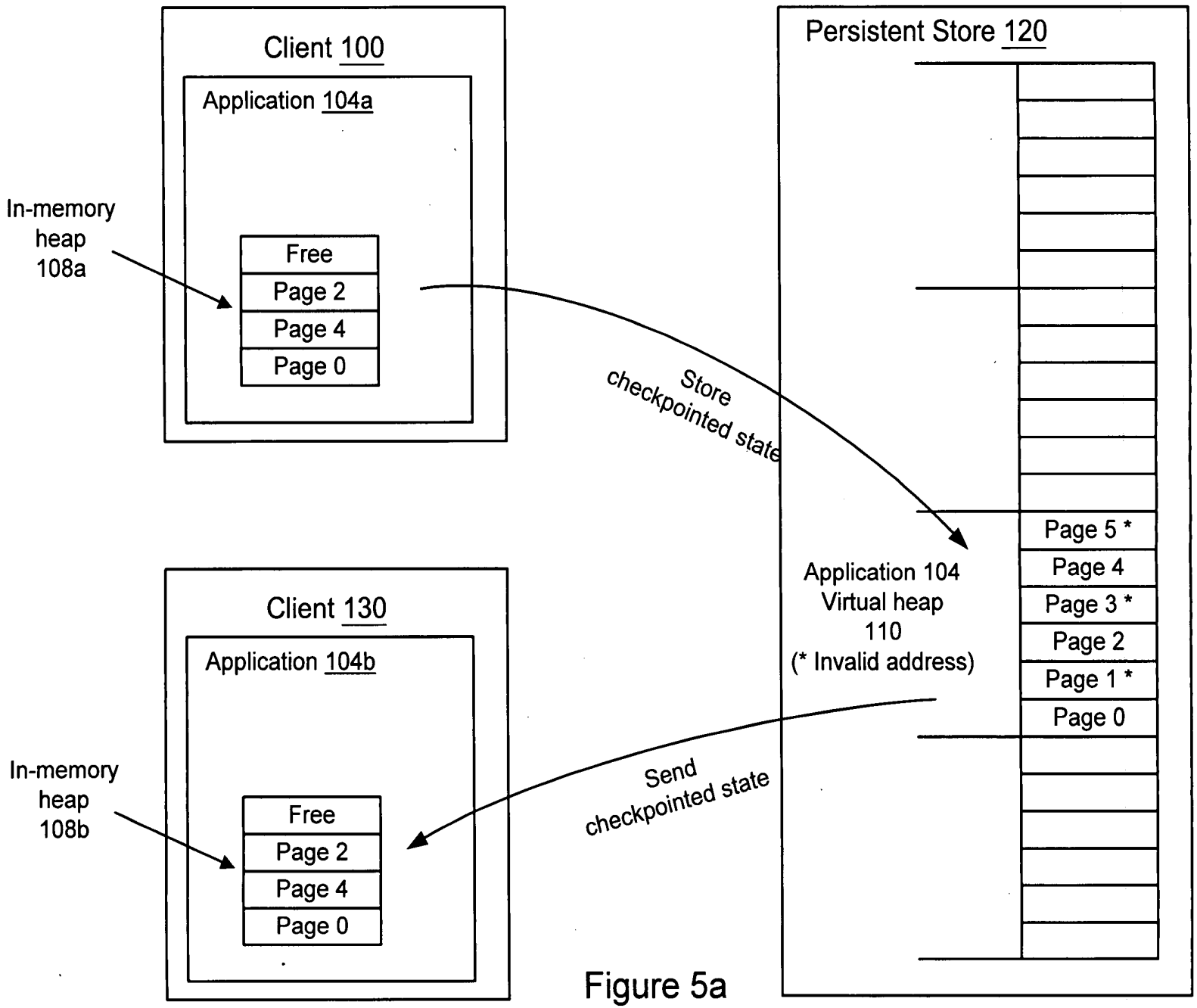
Client 100
Application 104a
In-memory heap 108a
Free
Page 2
Page 4
Page 0
Store checkpointed state
Persistent Store 120
Page 5 *
Page 4
Page 3 *
Page 2
Page 1 *
Page 0
Application 104 Virtual heap 110 (* Invalid address)
Client 130
Application 104b
In-memory heap 108b
Free
Page 2
Page 4
Page 0
Send checkpointed state

Figure 5a

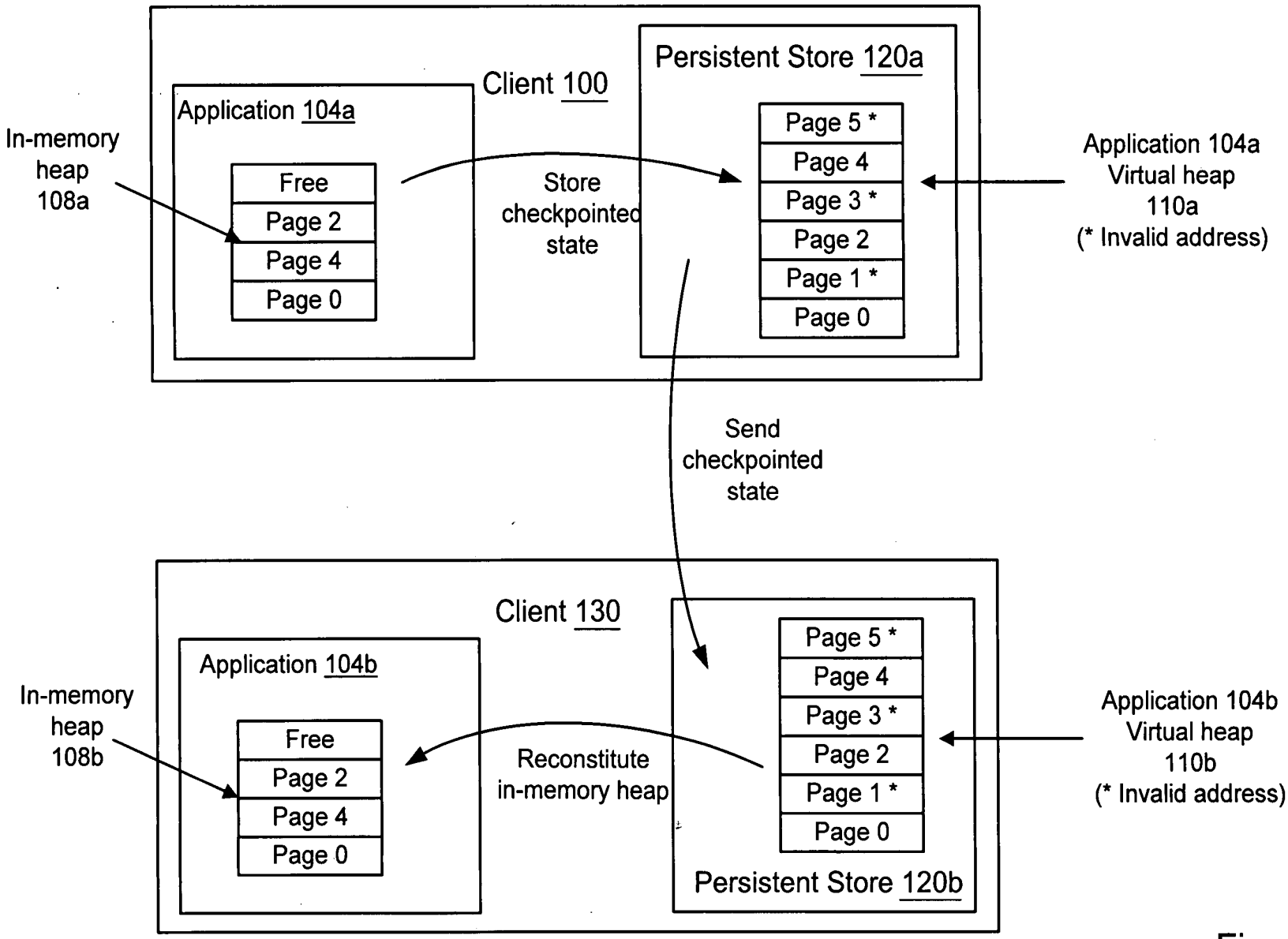
Client 100
Application 104a
Free
Page 2
Page 4
Page 0
In-memory heap 108a
Store checkpointed state
Persistent Store 120a
Page 5 *
Page 4
Page 3 *
Page 2
Page 1 *
Page 0
Application 104a Virtual heap 110a (* Invalid address)
Send checkpointed state
Client 130
Application 104b
Free
Page 2
Page 4
Page 0
In-memory heap 108b
Reconstitute in-memory heap
Persistent Store 120b
Page 5 *
Page 4
Page 3 *
Page 2
Page 1 *
Page 0
Application 104b Virtual heap 110b (* Invalid address)

Figure 5b

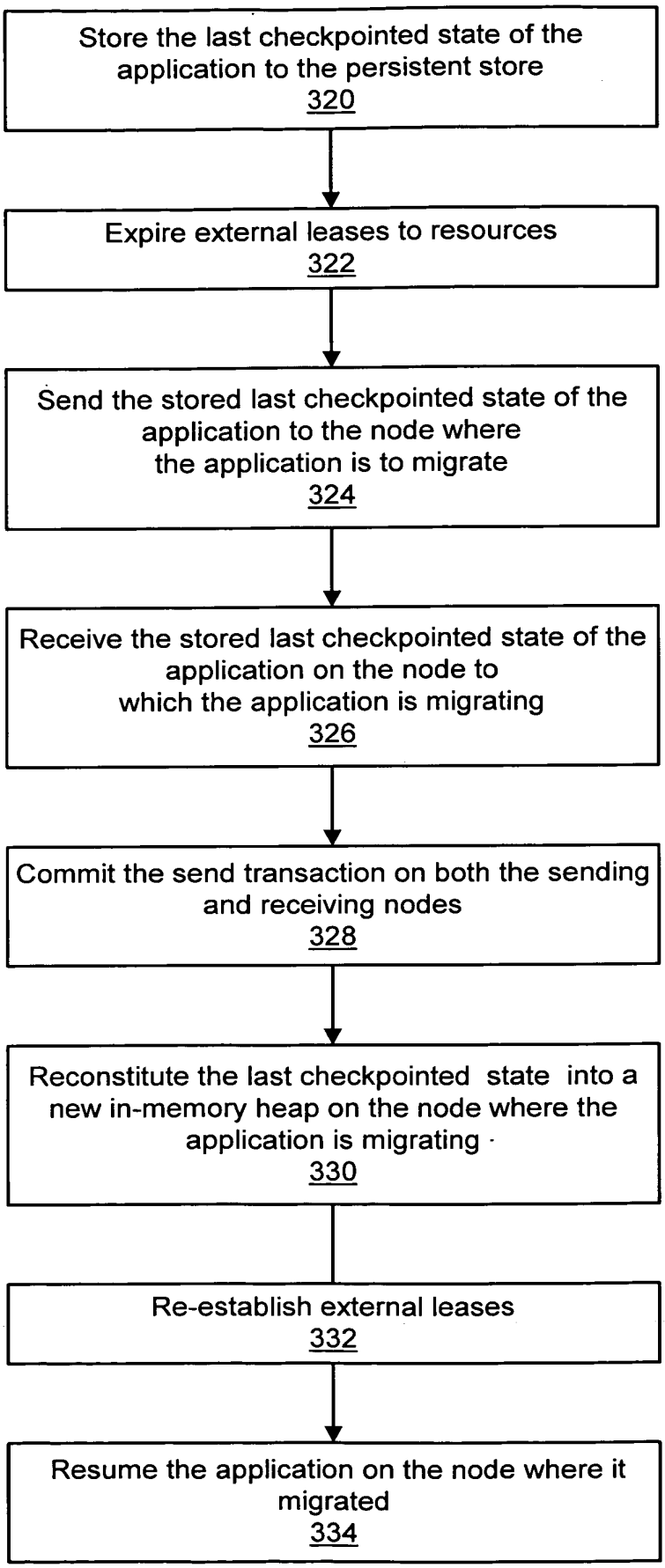
Store the last checkpointed state of the application to the persistent store
320
Expire external leases to resources
322
Send the stored last checkpointed state of the application to the node where the application is to migrate
324
Receive the stored last checkpointed state of the application on the node to which the application is migrating
326
Commit the send transaction on both the sending and receiving nodes
328
Reconstitute the last checkpointed state into a new in-memory heap on the node where the application is migrating
330
Re-establish external leases
332
Resume the application on the node where it migrated
334

Figure 6

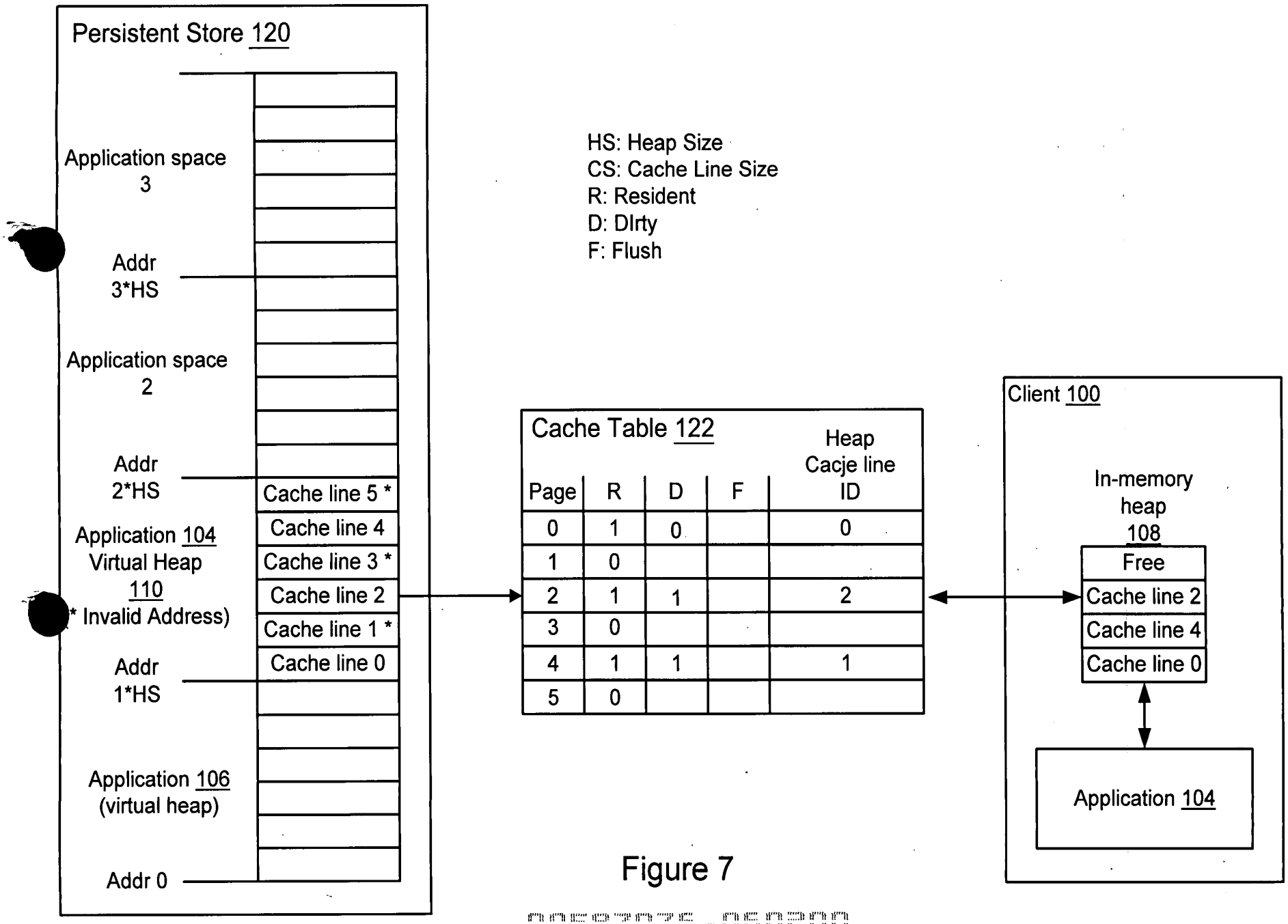
Persistent Store 120
Application space 3
Addr 3*HS
Application space 2
Addr 2*HS
Cache line 5 *
Cache line 4
Cache line 3 *
Cache line 2
Cache line 1 *
Cache line 0
Application 104 Virtual Heap 110
* Invalid Address)
Addr 1*HS
Application 106 (virtual heap)
Addr 0
HS: Heap Size
CS: Cache Line Size
R: Resident
D: DIrty
F: Flush
Cache Table 122
Page
R
D
F
Heap Cacje line ID
0 1 0 0
1 0
2 1 1 2
3 0
4 1 1 1
5 0
Client 100
In-memory heap 108
Free
Cache line 2
Cache line 4
Cache line 0
Application 104

Figure 7

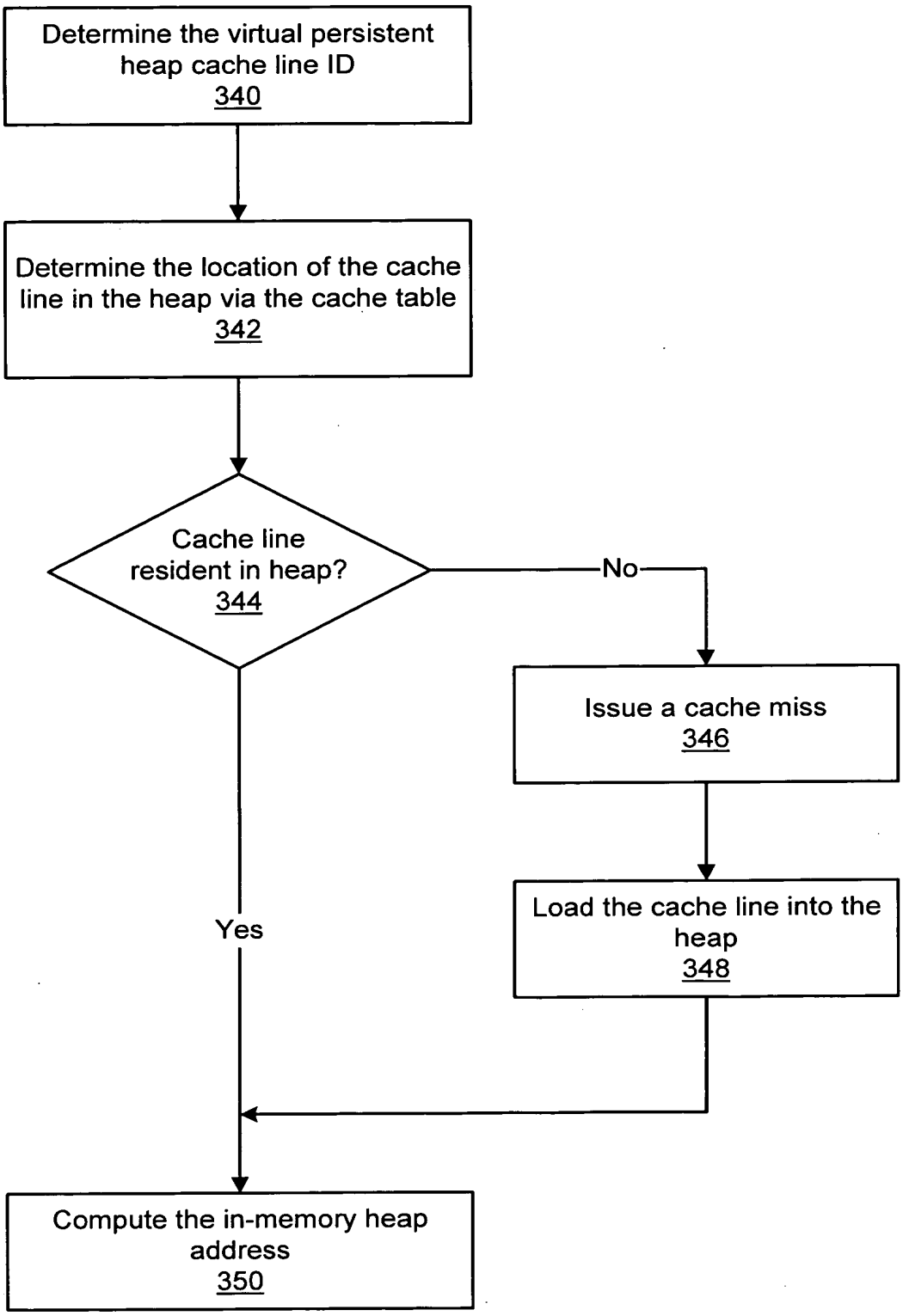
Determine the virtual persistent heap cache line ID
340
Determine the location of the cache line in the heap via the cache table
342
Cache line resident in heap?
344
No
Issue a cache miss
346
Load the cache line into the heap
348
Yes
Compute the in-memory heap address
350

Figure 8

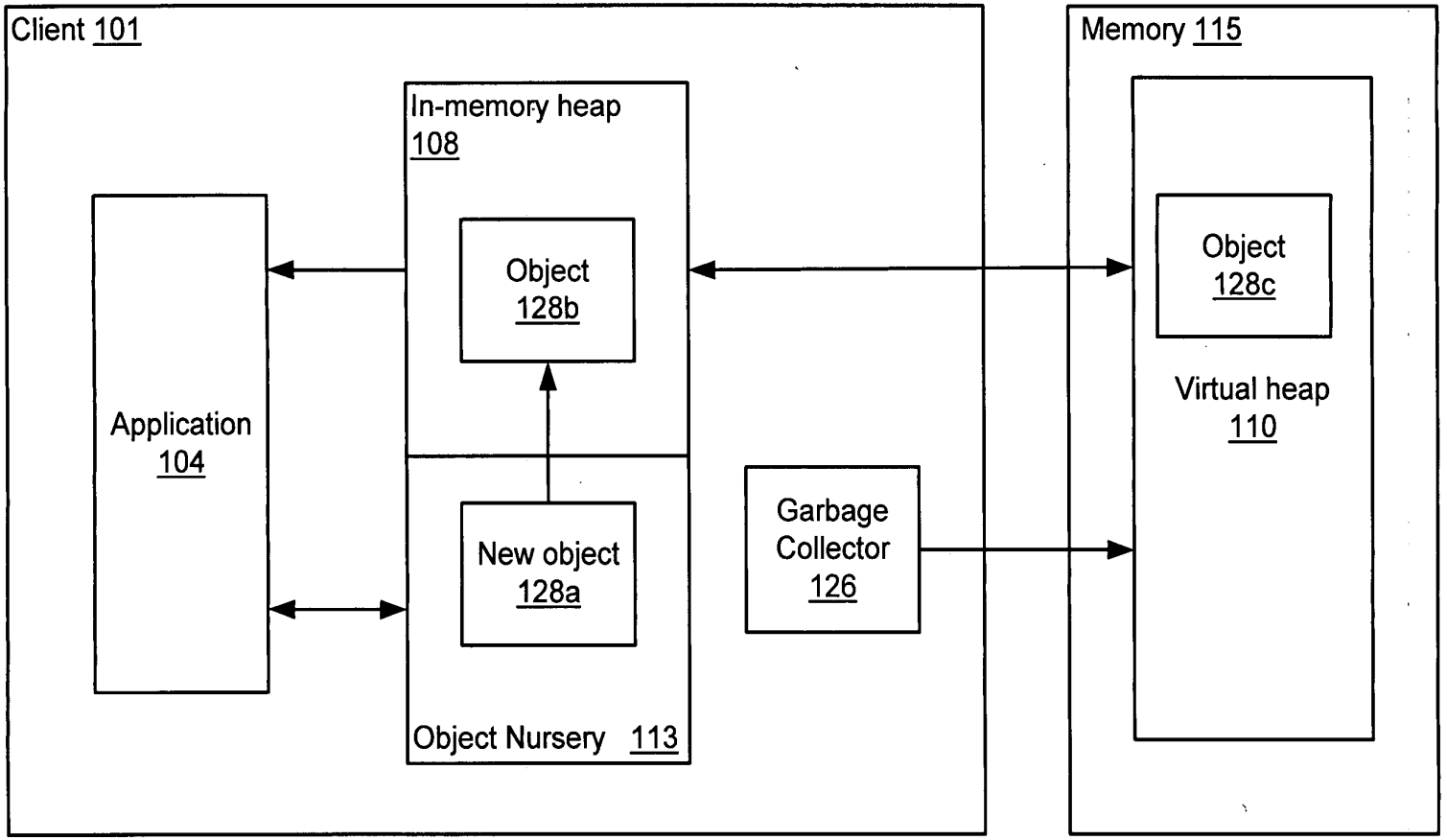
Client 101
In-memory heap
108
Object
128b
Application
104
New object
128a
Object Nursery 113
Garbage
Collector
126
Memory 115
Object
128c
Virtual heap
110

Figure 9

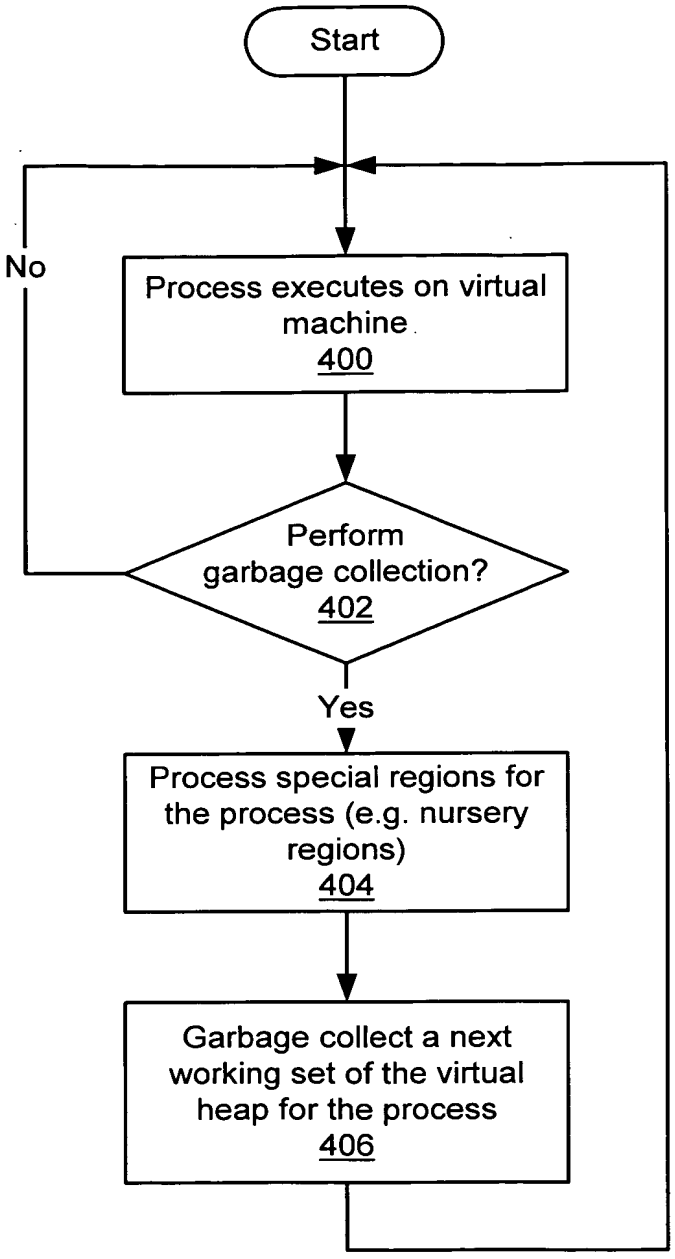
Start
No
Process executes on virtual machine
400
Perform garbage collection?
402
Yes
Process special regions for the process (e.g. nursery regions)
404
Garbage collect a next working set of the virtual heap for the process
406

Figure 10a

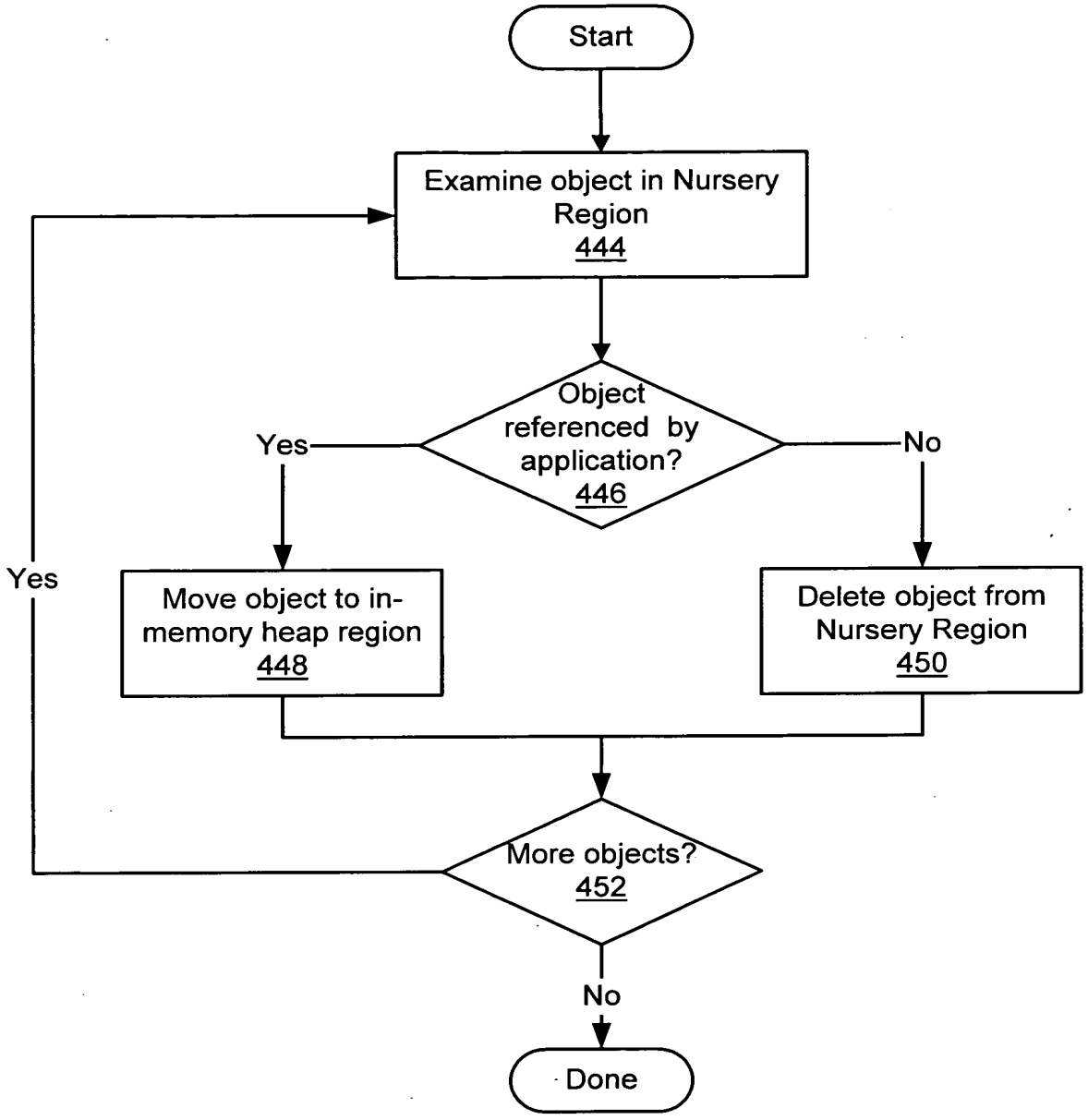
Start
Examine object in Nursery Region
444
Object referenced by application?
446
Yes
No
Move object to in-memory heap region
448
Delete object from Nursery Region
450
More objects?
452
Yes
No
Done

Figure 10b

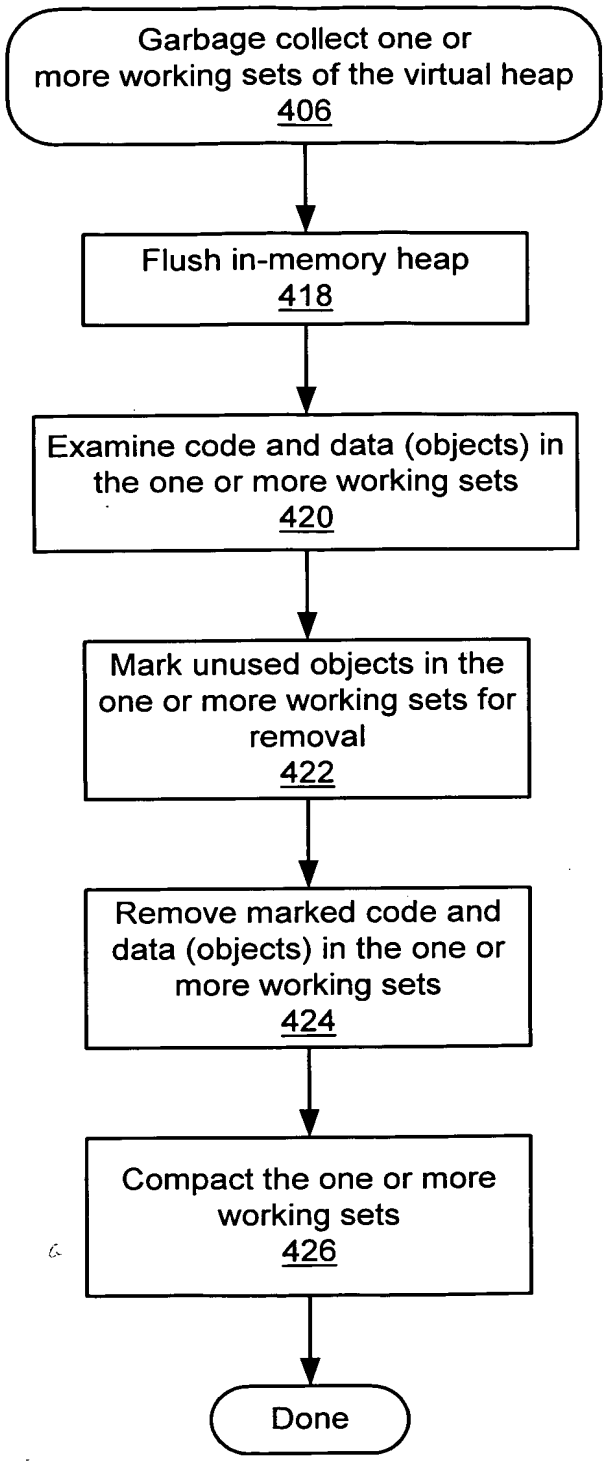
Garbage collect one or more working sets of the virtual heap
406
Flush in-memory heap
418
Examine code and data (objects) in the one or more working sets
420
Mark unused objects in the one or more working sets for removal
422
Remove marked code and data (objects) in the one or more working sets
424
Compact the one or more working sets
426
Done

Figure 10c

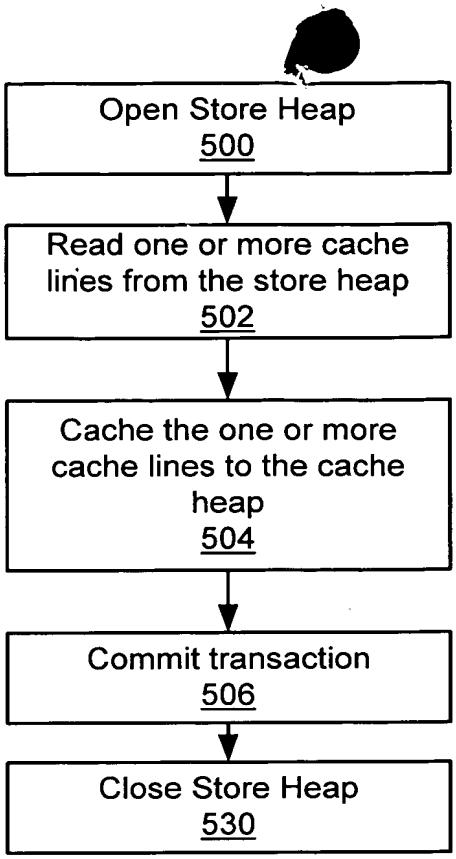

Figure 11a

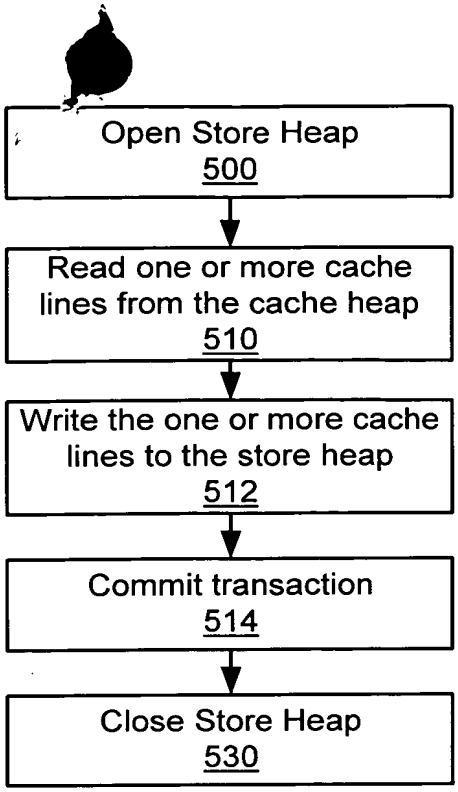

Figure 11b

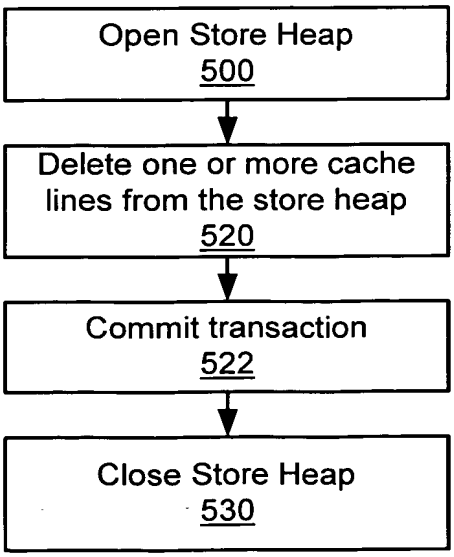

Figure 11c